Minimization of Timing Failures in Pipelined Designs via Path Shaping and Operand Truncation

Ioannis Tsiokanos, Lev Mukhanov, Dimitrios S. Nikolopoulos and Georgios Karakonstantis
Institute of Electronics, Communications and Information Technology (ECIT), School of EEECS, Queen's University Belfast
Email: {itsiokanos01, l.mukhanov, d.nikolopoulos, g.karakonstantis}@qub.ac.uk

***Abstract*—The continuous scaling of transistor sizes and the increased static and dynamic parametric variations render nanometer circuits more prone to timing failures. To protect circuits from such failures, typically designers adopt pessimistic timing margins, which are estimated statically under rare worst-case conditions. In this paper, we aim at minimizing the timing failures, while avoiding such pessimistic margins by proposing an approach that initially minimizes the number of long latency paths within each processor pipeline stage and constraints them in as few stages as possible. Such an approach, not only reduces the timing failures, but also limits the potential error prone locations to only few pipeline registers/stages. To further reduce these failures, we exploit the path excitation dependence on data patterns and we truncate the bit-width of the operands in the few remaining long latency paths by setting a number of least significant bits to a constant value zero. Such truncation may incur quality loss, but this can be controlled by carefully selecting the number of truncated bits and will be in any case less than the catastrophic loss that may be incurred under random timing failures. Additionally, our framework performs post-place and route dynamic timing analysis based on real operands that are extracted from a variety of applications, helping to estimate the dynamic timing failures, while considering the data dependent path excitation. When applied to an IEEE-754 compatible double precision Floating Point Unit (FPU), the proposed approach reduces the timing failures by 104.5× on average compared to a reference FPU design under an assumed 8.1% variation-induced worst-case path delay increase in a 45 nm process. Finally, results show that path shaping alone introduces an insignificant 0.25% area and 5.7% power overhead with no performance cost. The combination of path shaping with aggressive operand bit-width truncation leads to up-to 44.7% on average power savings due to the substantially reduced switching activity at a minimal quality loss.**

I. Introduction

Intensive transistor shrinking has worsened static and dynamic [1] process variations leading to 25% delay variations [2] and 20x spread in leakage [3] in sub-45nm technologies. The impact of such variations is on making circuits prone to timing failures and difficult to meet the power and performance specifications.

A. Conventional Variation-Aware Designs

To maintain high yield, manufacturers tend to adopt timing guard-bands that force the circuit to operate at a lower frequency or a higher voltage, hence providing sufficient timing margins to mitigate any failures triggered by delay variations [3], [4]. However, such timing margins are considered to be overly pessimistic since they are estimated based on few worst-case critical paths and on assumed rare operating conditions (e.g. worst-case temperature and operands), thus incurring large overheads [4], [1], [2] especially for the rest of the design (i.e. the inherently faster paths).

In an attempt to trim down the introduced overheads, statistical static timing analysis (SSTA) tools may have been introduced [5], but such tools still focus on improving the analysis and margin estimations rather than the design itself. Design-centric techniques focus on integrating extra circuits to detect any errors and either try to correct them in-situ using special flip-flops [6] or stall the pipeline and replay the failed instructions [7], [8]. Other design-centric schemes try to predict the instructions and operands that may activate the failure prone long latency paths (LLP) [9] and provide extra clock cycles for the completion of these paths. However, in all cases the enforced timing constraints for timing-error detection and prediction and the overheads for the applied recovery methods (e.g. multi-cycle replay penalties), especially in case of high activation probability of critical paths can offset the gains achieved by removing the static or dynamic safety margins.

An approach in [10] helps to limit the overheads of the above methods by reducing the overall number of many failure prone critical paths. Although effective, such a method has never been applied to a fully pipelined design and was not considered jointly with the above design-centric schemes or with methods for reducing the dynamic excitation of critical paths. Recently, schemes that take advantage of the error resilient properties of applications to tolerate a number of errors or approximate a number of less critical operations have also been proposed [11], [12], [13], [14], [15] for minimizing the traditional overheads. Nevertheless, the majority of such schemes were applied to individual digital signal processing (DSP) accelerators or arithmetic units rather than on pipelined designs. Furthermore, the majority of the approximate schemes use error-injection models based on simulators, without proper estimation of the potential impact of the reduced precision operands on the dynamic path excitation. More lately, approaches tried to limit the static or dynamic margins and dynamically adjust the clock in pipelined cores by considering the instruction and operand dependent path excitation [16], [17]. Such schemes may have shown the overly pessimistic estimation of timing paths by the conventional static timing analysis (STA) compared to the suggested use of dynamic timing analysis (DTA), however the proposed adjustment of the clock per instruction may be very challenging to be applied in practice.

B. Contributions and Outline

The primary aim of this paper is to minimize the potential timing failures in a pipelined design by appropriately redesigning the target circuit for limiting the failure-prone *LLP* and reducing the path excitation probability by truncating the operand bit-width. Our approach can effectively reduce the overheads of the traditional variation-aware schemes, while complementing existing design-centric and approximation-based schemes. The contributions of this work can be summarized as follows:

- We develop a framework based on state-of-the-art tools for redesigning a pipelined circuit by carefully shaping the computational paths, such that the *LLP* are significantly reduced and isolated to as few pipeline stages as possible. Such an approach not only reduces the timing failure probability, but also limits the potential failure prone locations to only few pipeline registers/stages, facilitating the use of any variation-aware or approximation scheme at low cost.
- We exploit the data dependent excitation of timing paths by truncating the bit-width of the operands that may activate the few remaining *LLP*. This is realized by setting a number of least significant bits (LSBs) of these operands to a constant value zero. By doing so, the computation delay of the *LLP* is reduced along with the excitation probability of these paths. In addition, the power consumption may also be reduced

due to the lower switching activity. It is also important to note that the operand bit-width truncation may lead to a deterministic quality loss, but this can be controlled by properly selecting the amount of the applied bit-width truncation and is expected to be less than the loss incurred by the random timing failures.

- We apply the proposed approach to the implementation of a variation-aware IEEE-754 compatible [18] FPU in a 45 nm process technology. Note that floating point (FP) variation-aware designs have not received much attention apart from few works [19], [20], despite the importance of FPUs in today's high-end processors. By combining a set of constraints at the synthesis and place and route phases with micro-architectures changes, we show that the *LLP* can be significantly reduced compared to an original FPU designed using conventional performance-centric optimizations.
- We develop a post-layout DTA tool to estimate the dynamic excitation of timing paths and potential timing failures by considering the operands of executed program traces and the clock period. The program traces and FP operands used as input to the DTA tool are extracted from real applications executed on a Reduced Instruction Set Computer (`RISC`) processor, helping to realistically evaluate our approach.
- We estimate the timing failures under various assumed clock reduction (CR) levels that represent potential variations for the proposed and original designs under different operand bit-width truncation ranges. Based on these estimations, we evaluate the quality loss quantified in terms of the signal-to-noise ratio (SNR) for three popular applications `Raytrace`, `CFD` and `Kmeans` from Computer Graphics, Fluid Dynamics and Data Mining domain, respectively.

The rest of the paper is organized as follows. Section 2 describes the proposed approach and the implemented flow using state-of-the-art tools. In Section 3, we apply the proposed framework to the design of an IEEE-754 compatible FPU of a `RISC` processor; and Section 4 presents the experimental results. Conclusions are drawn in Section 5.

II. Proposed Approach and Design Flow

Let us consider a pipelined design consisting of a set of N unique combinational paths $P = \{p_1, p_2, ..., p_N\}$, each requiring a delay $D(p_i)$ for $i = 1, 2, ..., N$ for completion. As in any synchronous design, the longest path across all S pipeline stages determines the clock period, such as:

$$CP_{STA} = \max_{\mathbf{s=1....S}} \left\{ \max_{\mathbf{p \in P^s}} \{D(p)\} \right\} = \max_{\mathbf{p \in P}} \{D(p)\}, \quad (1)$$

where P^s is the set of path groups that can be found uniquely in any of the s pipeline stages for $s = 1, 2..., S$.

Figure 1a depicts a typical distribution of all the path delays $D(P)$, which is obtained by applying conventional design flows and STA. As it can be seen, such a distribution is characterized by a so-called "timing wall", which is a consequence of how modern designs are optimized for power and area, subject to a frequency constraint: the *LLP* are optimized by gate up-sizing and buffering, while the inherently short latency paths (*SLP*) are allowed to become near-critical for recovering any area or power costs [10]. This "timing wall" has no negative impact on the adopted CP_{STA} of the design, however it critically affects the probability of timing failures since under any (even small) delay variation many paths may fail [21].

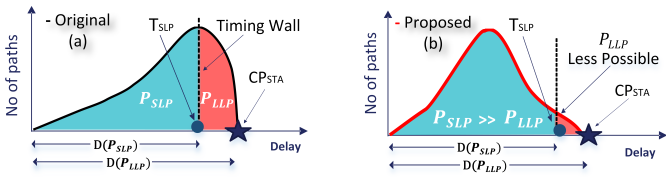


Fig. 1: Conventional vs proposed path distribution

Consider a set of K *SLP* as $P_{SLP} = \{p_1^{SLP}, ..., p_k^{SLP}\} \subset P$ and a set of operands Op_{SLP} that excite such paths and assume that all these paths can complete their computations within a time: $T_{SLP} = \max\{D(P_{SLP})\}$, as shown in Figure 1a. The same figure depicts a set of M *LLP*, namely $P_{LLP} = \{p_1^{LLP}, ..., p_M^{LLP}\} \subset P$, which is activated by another set of operands Op_{LLP}. The execution of the *LLP* can be completed within the CP_{STA}, but they require more time than T_{SLP}, i.e. $T_{SLP} < D(P_{LLP}) \leq CP_{STA}$. If at time instant t the executed instruction that is in the pipeline stage s and the relevant input operands activate only paths from P_{SLP}, then a positive timing slack $(CP_{STA} - D(p_i^{SLP}))$ can be observed. Such a slack can be used as a safety margin against potential delay variations, minimizing the probability of a timing failure at that stage. Intuitively, if during execution at most time instances only the *SLP* are being triggered across all stages, then the overall probability of a timing failure will be significantly reduced.

Path Shaping and Critical Stage Constraining. Therefore, the first step of our approach for reducing the timing failures is to move away from a path distribution with many *LLP* close to the CP_{STA}, which is typical in conventional designs (see Figure 1a). This sets essentially as a goal the minimization of $|P_{LLP}|$ subject to the design constraints such as the target clock period CP_{STA} (as well as the area and/or power). To achieve such a goal, we define appropriate timing constraints for different path groups and impose them on the design during synthesis. By introducing such path constraints, we ensure that the inherently fast paths *SLP* are not made slower as opposed to the conventional approach. The end goal is to obtain a path distribution similar to the one depicted in Figure 1b, where $|P_{SLP}| >> |P_{LLP}|$. Note that to facilitate the isolation of discerned path groups, we also make modifications at the micro-architectural or register-transfer level (RTL) by trying to constrain the *LLP* in as few stages as possible. This does not only help to control better the path groups, but also enables the isolation of P_{LLP} to as many stages as accessed by only few specific instructions. This allows us to limit the use of any failure avoidance or correction technique on fewer places (i.e stages) rather than on the overall design which is far more complicated and costly. In addition, it facilitates the development of failure avoidance/mitigation mechanisms tailored for the specific instruction(s) that activate the few remaining *LLP*. It is also important to note that our approach does not change the CP_{STA}, since any path shaping is made subject to maintaining the conventional speed.

Operand Truncation. The activation of each combinational path within each stage in a pipeline design depends on the executed instruction and on the input operands of each operation that takes place in each stage. Hence, it could be possible to further reduce the activation probability of the *LLP* by adjusting the operands of the specific instructions that excite these paths. A straightforward way that we suggest is to set to "0" some LSBs of Op_{LLP}, truncating effectively the bit-width and thus essentially reducing the computational delay [13]. Let us consider a simple example of adding two n bit numbers to illustrate the concept. Let A= 00.....0001 and B= 11.....1111, then the carry generated in the first bit position (i.e LSB) is propagated all the way to the final bit position, activating the error-prone *LLP*. Let us modify the inputs by inserting "0" in the last 4 bits, such as: A= 00..0..0000 and B= 11..1..0000. In this case there is no carry propagation across the middle bit and thus only the *SLP* will be excited [9]. Essentially, such a concept allows us to obtain a positive timing slack (since the circuit becomes faster) that can limit any potential risk of failure in case of any variation-induced delay increase.

TABLE I: FP addition with random (due to potential timing failures) and deterministic (due to deliberately setting bits to zero) errors

	Operand A (64 bits)	Operand B (64bits)	Result C (64bits)	MSE
Reference	0100000010001111001111000000101 0001111010111000010100111101100 (decimal: 999.7525)	0100000011010000000111101101110 0110011001100110011001100110110 (decimal: 16507.475)	0100000011010001000110001100111 1000111101011100001010001111010 (decimal: 17507.2275)	0
Random error Bit flip in the 2^{nd}bit	0100000010001111001111000000101 0001111010111000010100111101100 (decimal: 999.7525)	0100000011010000000111101101110 0110011001100110011001100110110 (decimal: 16507.475)	0000000011010001000110001100111 1000111101011100001010001111010 (decimal: $\sim 973872 * 10^{-310}$)	$306.5 \cdot 10^6$
Deterministic error truncating 32 LSBs	0100000010001111001111000000101 0000000000000000000000000000000 (decimal: 999.7524)	0100000011010000000111101101110 0000000000000000000000000000000 (decimal: 16507.468)	0100000011010001000110001100111 0010100000000000000000000000000 (decimal: 17507.2211)	$4.096 \cdot 10^{-5}$

The truncation of a number of LSBs from Op_{LLP} may provide a slack and reduce the LLP excitation as we discussed above, but this will come at a quality loss. However, such a loss can be controlled by appropriately selecting the number of truncated LSBs, ensuring that it is not as catastrophic as the loss that can be incurred by the random timing failures in case they affect the most significant bits (MSBs). For instance, let us assume two floating numbers as operands A and B, whose addition results to C. As illustrated in Table I, a bit flip in the 2^{nd} bit of the result (highlighting in red) due to timing failure, will lead to a completely different result with a solid mean square error (MSE) compared to the reference (MSE: $306.5 \cdot 10^6$). Conversely, the deterministic error due to the truncation of the 32 LSBs of each operand (truncated bits in red) is insignificant (MSE: $4.096 \cdot 10^{-5}$). Bit-width truncation can also save power, since it directly reduces the switching activity.

Exploiting the Dynamic Path Excitation. An essential difference between our approach and the conventional one, which relies only on STA is that we exploit the dynamic excitation of paths by operands. The fact that the excitation of the LLP by Op_{LLP} is made rare by design minimizes the need of any conservative margin. Nonetheless, in case of Op_{SLP} there is enough positive timing slack to avoid failures under any potential worst-case path delay increase up to a magnitude of $\Delta T_{maxvar} \leqslant (CP_{STA} - T_{SLP})$.

To estimate the efficacy of our approach and evaluate the dynamic data dependent excitation of paths, we develop a tool to perform DTA. Such a tool aims at uncovering the unused timing margins of the processor that are available at runtime, which cannot be accurately characterized by STA due to the missing notion of path activation probabilities. Additionally, with this analysis, we also estimate how often the LLP are excited and the quality degradation incurred by the operand truncation. Finally, such an analysis phase helps to extract instruction aware timing failures, which also depend on the dynamic excitation of critical paths by operands.

A. Design Flow

The steps of the proposed approach are implemented using state-of-the-art electronic design automation (EDA) tools and our modifications are highlighted in orange in Figure 2. As can be seen, our flow consists of a design phase and an analysis phase.

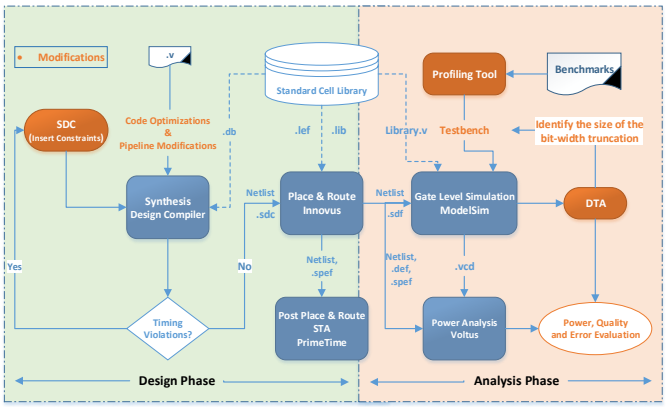


Fig. 2: Workflow of the proposed approach

1) Design Phase: To reduce $|P_{LLP}|$, as we discussed above, we impose constraints on different path-groups (in the Synopsys Design Constraints or SDC file) based on the minimum delay required for the completion of each path-group. By introducing multiple path-group constraints, we force the synthesis tool to avoid optimizations that make naturally fast paths slower for saving area/power. These constraints may have an impact on the area and power consumption, but the resulting overheads that depend on the targeted path distribution, can be kept small. Initially, we apply strict constraints to shift the "timing wall" away from the target CP_{STA}. If there are timing violations after synthesis, we relax the design constraints and re-run our iterative method until the timing target is met. Furthermore, to restrict P_{LLP} to one stage, we perform micro-architectural changes, while ensuring that the target clock frequency of the design is still met. The synthesis step is followed by the place and route using the Innovus Tool from Cadence. Sign-off STA will follow with Synopsys PrimeTime to verify that design has achieved the timing closures.

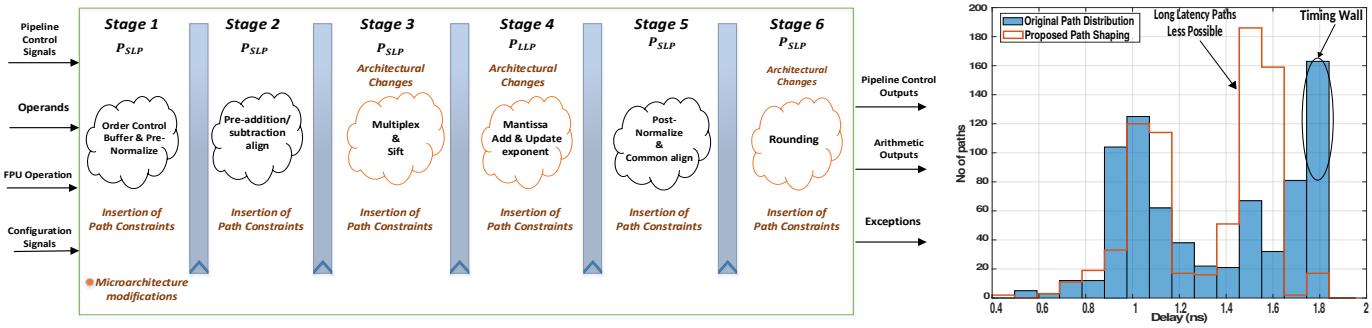
2) Analysis Phase: **DTA**. To enable characterization of the data dependent path activation, we use the post-place gate-level simulation supported by ModelSim, which monitors the inputs and the outputs of all flip-flops in the design and generates a corresponding event log. To obtain this information and perform full back annotated simulation, ModelSim apart from RTL netlist and testbench requires a standard delay format (SDF) file which describes the cell and interconnect delay. RTL netlist and SDF file are obtained by the place and route step, while our profiling tool feeds the ModelSim with real FP operands, creating essentially the required testbench. Providing that every set of operands under nominal conditions produces an error-free output, we define this error-free gate-level simulation output as D_{gold}.

To measure the number of manifested timing failures under any potential delay increase, we execute the simulation and compare the D_{gold} with the event log obtained from ModelSim. Finally, this tool can extract a value change dump (VCD) file that contains information about the switching activity and value changes that occurred during the simulation for nets and registers of the design. This file is essential for performing dynamic power analysis.

Power Analysis. We estimate the power consumed by all benchmarks and designs using the Voltus tool from Cadence. To perform dynamic power analysis, we give the following inputs to the Voltus: the post placed and routed netlist, the VCD file, a design exchange format (DEF) file that represent the physical layout and a standard parasitic exchange format (SPEF) file which corresponds to the parasitic data of wires in a chip. Sign-off quality power analysis obtains VCD files from ModelSim, while the other inputs are produced by the already placed and routed design.

III. Case Study: Application to FPU

We apply the proposed approach to a multi-cycle, IEEE-754 compatible double precision FPU. This representative pipelined architecture follows the representation: $-1^S \times M \times 2^E$, where S : sign, E : exponent and M : mantissa. In a double precision FP



(a) (b)

Fig. 3: (a) Proposed micro-architecture and (b) path distribution of the FPU (modifications are emphasized in orange)

number defined by the IEEE-754 Standard, the MSB indicates the sign, the next 11 bits represent the exponent and the mantissa consists of the 52 LSBs. This FPU is part of the latest Out-of-Order mor1kx MAROCCHINO pipeline, a 5-stage pipeline microprocessor based on the `OpenRISC` 1000 Instruction Set Architecture [22]. In this paper, we implement the following FP instructions: addition/subtraction, integer to FP and FP to integer conversions and finally, comparison between FP numbers. Figure 3a illustrates the micro-architecture of the targeted FPU, highlighting the FP addition/subtraction. At Stage 1, an Order Control Buffer and a Pre-Normalize block are implemented, which permits data dependencies detection and adjustment of the exponent and mantissa, respectively. Stage 2 is responsible for the pre-addition/subtraction alignment, while Stage 3 performs the necessary multiplexing and shifting of the operands. Mantissa addition and exponent update are performed at Stage 4; rounding occurs in the last two stages.

A. Redesigned FPU

We start by applying the typical EDA flow (see Figure 2) to the conventional unmodified design. After following the synthesis and place and route steps, as well as performing STA using PrimeTime, we built the path distribution that is shown in Figure 3b. The obtained distribution implies that the conventional performance-centric flow results in large $|P_{LLP}|$, in which many paths are close to the worst case delay, or in other words to the clock delay. As it was discussed, such a path distribution creates the "timing wall". Figure 4a depicts the path distribution within each pipeline stage, revealing that the "timing wall" exists in 4 out of the 6 stages. These findings indicate that there is an increased possibility of timing failures in the stages where *LLP* exist. To circumvent this, we apply the steps of our design flow with the next modifications:

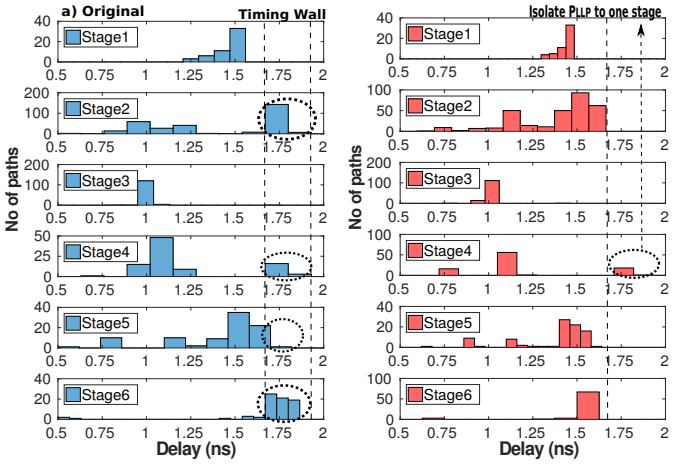


Fig. 4: Original and proposed path distribution across stages

Micro-architectural changes. To change the distribution of paths in the manner discussed in Section II, we apply micro-architectural modifications to some pipeline stages. In particular, after performing STA, we noticed that Stage 4 consists of the most timing critical paths. To reduce the number of the *LLP* in this stage, we moved parts of the combinational logic to the previous stage, exploiting the slack margins observed at Stage 3. Additionally, rounding, which occurs at Stages 5 and 6, poses a bottleneck because it is applied on the result of all FP operations and thus many paths in these stages are long latency ones. To this end, we re-write part of the RTL code in Stage 5 and 6, making the synthesis part more efficient. Specifically, we modify some necessary conversions to negative numbers such that large conversions for rarely covered ranges are eliminated.

Group path constraints. As mentioned in Section II.A.1, during the synthesis step, we apply various constraints by grouping paths at different stages based on their latency. We also constrain the *LLP* to as few stages and instructions as possible. As a result, we reduce $|P_{LLP}|$ (see Figure 3b) and isolated it to Stage 4, as can be observed in Figure 4b, while ensuring all other paths are fast enough (at least 8.5% faster than CP_{STA}) to minimize timing failures caused by delay variations. It is also worth mentioning that the remaining *LLP* are isolated in such way that can be triggered only by FP addition/subtraction instructions in the Stage 4, which was also one of the main goal of the design strategy.

B. Application of operand truncation

After redesigning the FPU, we apply bit-width truncation to the input operands of the specific instructions that were found to activate the few remaining *LLP* in order to reduce their excitation probability and thus the timing failures. Since all the *LLP* are restricted to Stage 4 of the FP addition/subtraction instructions, we deploy the truncation only to the LSBs of the input operands of these instructions. Given that Stage 4 essentially implements the 52-bit mantissa addition, we set constant "0" values to the LSBs to avoid delay failures in MSBs.

As we mentioned in Section II, the truncation may lead to quality loss and needs to be carefully selected. In our experiments, we selected to evaluate the truncation of the 32, 44 and 48 LSBs of the mantissa part of the FP addition/subtraction operands. This means that the sign and the exponent part of the IEEE-compliant operands were unaffected (12 bits totally) along with the 20, 8 and 4 MSBs of the mantissa part in the considered scenarios.

TABLE II: Clock reduction levels (worst-case delay increase)

CR	CR0	CR1	CR2	CR3	CR4	CR5
clock, (ns)	1.85	1.80	1.75	1.7	1.65	1.55
reduction, (%)	0	2.7	5.4	8.1	10.8	16.2

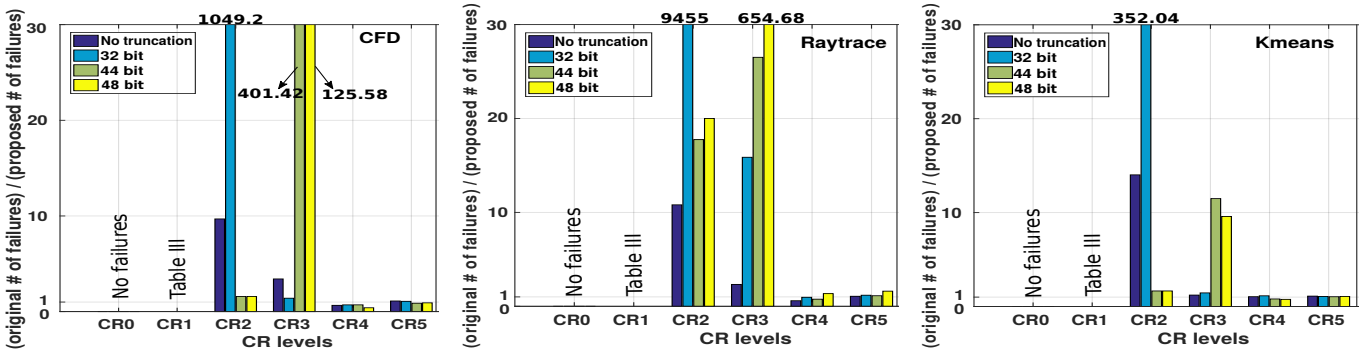


Fig. 5: Timing failures under various CR and operand truncation levels across 3 benchmarks (caption shows the number of the truncated bits)

IV. Evaluation Results

In this Section, we evaluate the efficacy of our approach in limiting the timing failures under various CR levels, which are shown in Table II. The different CR levels represent potential degrees of worst-case path delay increase that may be caused by variations [23]. To feed the DTA tool with real FP operands, we developed a profiling tool based on [24]. Using such a tool, we extract program traces after running various compute intensive applications from Data Mining, Fluid Dynamics and Computer Graphics domains. In particular we use the `Kmeans` and `CFD` benchmarks from the Rodinia suite; and the `Raytrace` benchmark from the Parsec suite. We run the profiler on `ARM A7 RISC` cores and we observed that FP instructions are responsible for about 32% on average of the total instructions across all benchmarks. Then, we extract 10.000 operands from the most frequently executed FP instructions for each application, which are used as input to the DTA tool for measuring the timing failures and estimating the quality loss in terms of SNR, as well as for estimating the power consumption. The FPU is implemented using the design flow described in Section II.A in NanGate 45 nm Composite Current Source Cell Library [25]. In the rest of this Section, we compare our redesigned FPU (explained in Section III.A) with the original (reference) design. For fair comparison, we applied the truncation of 32, 44 and 48 LSBs of specific FP operands to the or(iginal) FPU as well to the pr(oposed) one.

A. Characterization of Timing Failures

Figure 5 demonstrates how the number of timing failures changes under different CR and bit-width truncation levels when we scale down the clock period in the original and the proposed design across the 3 benchmarks. Timing failures are a function of input operands, clock period and the number of truncated bits and are reported only in case that the simulated output for specific operands differs from the recorded error-free output D_{gold}. As shown in Figure 5, under the nominal clock period (CR0) no failures are manifested. Moreover, we observe that the timing failures are substantially decreased under any degree of bit-width truncation in case of all applications. In particular, by setting to zero the 32, 44 and 48 LSBs of the mantissa in the original FPU, the total number of failures across the 3 benchmarks has been decreased by 1.69×, 2.83× and 12.13× on average.

In the same figure, we can observe that the number of timing failures incurred in the original design is significantly larger than the number of failures incurred in the proposed FPU under CR1, CR2 and CR3. Beyond the CR3 level that essentially corresponds to the ΔT_{maxvar}, we notice an increase in timing failures, especially in the proposed design. This can be attributed to the fact that the applied path shaping have shifted the paths as shown in Figure 3b, resulting in a high number of paths being activated and failing under CR4 and CR5. In particular, Figure 3b shows that there are more paths with delay of 1.4 ns - 1.65 ns in the proposed design than in the original one; this outcome implies that the probability of timing violations in case of activation of these paths under CR4 and CR5 is expected to be higher in our design.

TABLE III: Timing failures under CR1 and various truncation level

no of truncated bits	0	32	44	48
CFD (or , pr)	107505 , 0	24508 , 0	86 , 0	86 , 0
Raytrace (or , pr)	79985 , 0	38 , 0	38 , 0	22 , 0
Kmeans (or , pr)	16058 , 0	78 , 0	78 , 0	78 , 0

Nonetheless, the proposed design results in a significantly reduced number of timing failures for most of the CR levels. Especially, our approach under CR1 allows us to eliminate all the incurred errors, while as we can see in Table III the original FPU exhibits from 86 up-to 107505 failures for the same CR. In addition, we can observe that the proposed design limits considerably the timing failures under the CR1, CR2 and CR3 levels. In particular, under CR3, which corresponds to an 8.1% worst-case path delay increase, the proposed design with the preferred path shaping reduces timing failures by 2.32× on average compared to the original. The combination of path shaping and operand bit-width truncation in the 32, 44 and 48 LSBs reduces these failures by 6.24×, 146.48× and 263.28× on average, respectively when compared to the original design.

B. Evaluation of Quality

To evaluate the quality loss incurred by the operand truncation and the random timing failures, we estimate the SNR of our design and compare it with the SNR of the original FPU in case of all three applications. SNR compares the level of a desired signal to the level of the "noise" incurred either by timing failures or bit-width truncation. In our case the desired signal is the D_{gold}, while the "noise" is estimated by the MSE between the D_{gold} and the system output under a specific CR and bit-width truncation level. The effect of different CR levels and operand bit-width truncation on the SNR before and after applying our approach is depicted in Figure 6. In case of `CFD`, we can observe that the SNR of the proposed FPU up-to CR3 and under different levels of truncated bits is much better and even close to the reference D_{gold} signal without noise (i.e SNR of the original double precision FPU at CR0). Under CR4 and CR5 the SNR levels in both the original and the proposed designs are very low and unacceptable. In case of `Raytrace`, results are similar, where SNR in the proposed design under CR3 is up-to 1.4× better compared to the original one. In case of `Kmeans`, we observe that the proposed design exhibits SNRs ranging from 40db till 280db up-to CR3 under no or 32 LSBs truncation, while the original one presents very low or even negative SNRs. Note that if we increase the number of truncated bits by more than 42 in both the original and the proposed designs, the output quality degrades significantly. This

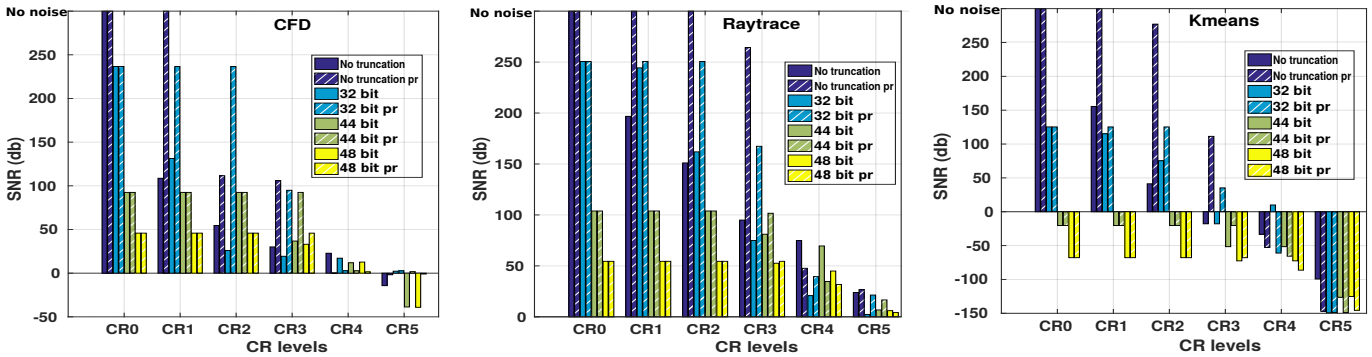


Fig. 6: SNR under various CR and operand bit-width truncation levels across 3 benchmarks (caption shows the number of the truncated bits)

can be explained by the large values of FP operands in `Kmeans` with mean value $3.75*10^6$, which is considerably greater than the mean value of the operands in `Raytrace` and `CFD` ($9.9*10^{-5}$ and 1.65, respectively). This partially explains why `Raytrace` and `CFD` show less quality loss in case of aggressive bit-width truncation.

Overall, we found that setting the last 32 bits of mantissa to "0" provides a judicious choice for the operand truncation, considering all the evaluated applications. Finally, the limited incurred loss observed in the proposed approach can be attributed to the fact that the failing bits are neither in the exponent bits nor in the MSBs of the mantissa.

C. Estimation of Power and Area

Table IV depicts the differences of the power consumed by the post-placed and routed original and proposed design measured with the EDA tools, as explained in Section II.A.2. Our approach, when only path shaping is applied introduces a 5.7% power overhead. However, the proposed FPU leads to up-to 44.7% power saving when path shaping and 48 LSBs truncation are combined, due to the significantly reduced switching activity. Overall, the area overhead incurred by path shaping is 0.25%, while our approach does not incur any performance loss, which would otherwise be incurred by conventional timing margins.

TABLE IV: Average power consumption under various truncation levels

No of truncated bits	0 (or , pr)	32 pr	44 pr	48 pr
Power (mW)	24.71 , 26.12	19.99	16.91	13.66
Power difference (%)	0 , -5.7	19.1	31.6	44.7

V. Conclusion

This paper presented a framework for minimizing the timing failures in pipelined designs by i) redesigning the target circuit in a way that eliminates the excitation of the *LLP*, and ii) opportunistically exploiting the dynamic activation of such paths by few operands Op_{LLP} and making them rare by setting a constant value "0" to a fixed number of LSBs in the relevant operands. The evaluation of the proposed redesigned placed and routed FPU with the developed DTA tool using extracted program traces shows a significant reduction of timing failures under potential delay variations with a negligible 0.25% area overhead and no performance loss. Results also show that truncation of 32 or 48 LSBs of Op_{LLP} helps to maintain high SNR levels in all the evaluated applications and up-to an assumed 8.1% variation-induced worst-case delay increase. Finally, we observe that the path shaping may introduce 5.7% power overhead, but when combined with operand bit-width truncation can lead to up-to 44.7% power savings.

Acknowledgments

The presented research effort is partially supported by the European Community Horizon 2020 programme under grant no. 688540 (UniServer) and grant no. 732631 (OPRECOMP).

References

[1] W. Tschanz *et al.*, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE JSSC*, vol. 46, no. 1, pp. 194–208, Jan 2011.
[2] P. Gupta *et al.*, "Underdesigned and opportunistic computing in presence of hardware variability," *CAD*, vol. 32, no. 1, pp. 8–23, 2013.
[3] S. Borkar *et al.*, "Parameter variations and impact on circuits and microarchitecture," in *DAC*, June 2003, pp. 338–342.
[4] G. Karakonstantis *et al.*, "Containing the nanometer "pandora-box": Cross-layer design techniques for variation aware low power systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 1, pp. 19–29, 2011.
[5] C. S. Amin *et al.*, "Statistical static timing analysis: how simple can we get?" in *DAC*, 2005, pp. 652–657.
[6] Y. Zhang *et al.*, "irazor: Current-based error detection and correction scheme for pvt variation in 40-nm arm cortex-r4 processor," *IEEE JSSC*, vol. 53, no. 2, pp. 619–631, Feb 2018.
[7] D. Blaauw *et al.*, "Razor II: in situ error detection and correction for PVT and SER tolerance," in *ISSCC*, 2008, pp. 400–401.
[8] D. Bull *et al.*, "A power-efficient 32 bit arm processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation," *IEEE JSSC*, vol. 46, no. 1, pp. 18–31, Jan 2011.
[9] S. Ghosh *et al.*, "Voltage scalable high-speed robust hybrid arithmetic units using adaptive clocking," *IEEE TVLSI*, pp. 1301–1309, 2010.
[10] A. B. Kahng *et al.*, "Slack redistribution for graceful degradation under voltage overscaling," in *ASP-DAC*, 2010, pp. 825–831.
[11] G. Karakonstantis *et al.*, "Herqules: System level cross-layer design exploration for efficient energy-quality trade-offs," in *ISLPED*, 2010.
[12] V. K. Chippa *et al.*, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC*, 2013, pp. 1–9.
[13] K. Kunaparaju *et al.*, "Varot: Methodology for variation-tolerant dsp hardware design using post-silicon truncation of operand width," in *International Conference on VLSI Design*, 2011, pp. 310–315.
[14] G. Karakonstantis *et al.*, "Process-variation resilient and voltage-scalable dct architecture for robust low-power computing," *IEEE TVLSI*, vol. 18, no. 10, pp. 1461–1470, Oct 2010.
[15] J. Miao *et al.*, "Modeling and synthesis of quality-energy optimal approximate adders," in *ICCAD*, Nov 2012, pp. 728–735.
[16] J. Constantin *et al.*, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *DATE*, 2015, pp. 381–386.
[17] A. Rahimi *et al.*, "Application-adaptive guardbanding to mitigate static and dynamic variability," *Trans. on Computer*, pp. 2160–2173, 2014.
[18] IEEE 754-2008. IEEE 754-2008 Standard for Floating-Point Arithmetic.
[19] X. Liang *et al.*, "A process-variation-tolerant floating-point unit with voltage interpolation and variable latency," in *ISSCC*, 2008.
[20] S. Salehi *et al.*, "Energy and area analysis of a floating-point unit in 15nm cmos process technology," in *SoutheastCon*, April 2015, pp. 1–5.
[21] J. Patel, "Cmos process variations: A critical operation point hypothesis," April 2008 [Online].
[22] OpenRISC Community, "OpenRISC 1000 architecture manual.".
[23] X. Jiao *et al.*, "Slot: A supervised learning model to predict dynamic timing errors of functional units," in *DATE*, 2017, pp. 1183–1188.
[24] L. Mukhanov *et al.*, "Alea: Fine-grain energy profiling with basic block sampling," in *PACT*, Oct 2015, pp. 87–98.
[25] NanGate FreePDK45 Open Cell Library, http://nangate.com.